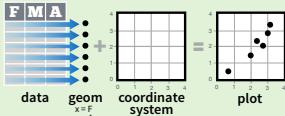


Data Visualisatie met ggplot2

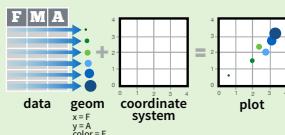
Spiekbriefje R Studio

De Basis

ggplot2 is gebaseerd op de **grafische grammatica**, het idee is dat je iedere grafiek kunt maken uit een klein aantal componenten: de **data set**, een set **geoms**—symbolen die datapunten voorstellen, en een **coördinaten systeem**.



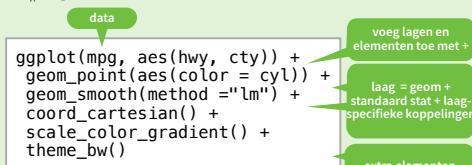
Koppel variabelen in de dataset aan de visuele kenmerken van de geom zoals **maat**, **kleur** en de **x** en **y** locaties om de data te tonen.



Maak de grafiek met **ggplot()** of **qplot()**

```
ggplot(data = mpg, aes(x = cty, y = hwy))
```

Hiermee begin je een grafiek waar je daarna lagen aan toe voegt. Kent geen standaardwaarden maar biedt meer mogelijkheden dan qplot().



Voeg een nieuwe laag toe aan je grafiek met een **geom_***() of **stat_***() functie. Beiden leveren een geom, een set uiterlijke kenmerken, een standaard statistiek en een positie aanpassingen.

aesthetic mappings **data** **geom**

```
qplot(x = cty, y = hwy, color = cyl, data = mpg, geom = "point")
```

Levert een complete grafiek met de gegeven data, geom en koppelingen. Veel handige standaardinstellingen beschikbaar.

last_plot()

Ga terug naar de laatste grafiek

```
gsave("plot.png", width = 5, height = 5)
```

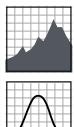
Bewaart de laatste grafiek als een 5'x5' bestand met de naam "plot.png" in de werkmap. Past het bestandstype aan aan de extensie.

Geoms - Gebruik een geom om datapunten en variabelen weer te geven. Elke functie resulteert in een laag.

Een Variabele

Continu

```
a <- ggplot(mpg, aes(hwy))
```



a + geom_area(stat = "bin")

x, y, alpha, color, fill, linetype, size

a + geom_density(kernel = "gaussian")

x, y, alpha, color, fill, linetype, size, weight

b + geom_density(aes(y = ..density..))

a + geom_dotplot()

x, y, alpha, color, fill



a + geom_freqpoly()

x, y, alpha, color, linetype, size

b + geom_freqpoly(aes(y = ..density..))

x, y, alpha, color, fill, linetype, size, weight

a + geom_histogram(binwidth = 5)

x, y, alpha, color, fill, linetype, size, weight

b + geom_histogram(aes(y = ..density..))

Discrete

```
b <- ggplot(mpg, aes(fl))
```



b + geom_bar()

x, alpha, color, fill, linetype, size, weight

Grafische Primitieven

```
map <- map_data("state")
```

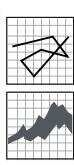
c <- ggplot(map, aes(long, lat))



c + geom_polygon(aes(group = group))

x, y, alpha, color, fill, linetype, size

```
d <- ggplot(economics, aes(date, unemploy))
```



d + geom_path(lineend = "butt", linejoin = "round", linemetre = 1)

x, y, alpha, color, linetype, size

d + geom_ribbon(aes(ymax = unemploy - 900, ymin = unemploy + 900))

x, ymax, ymin, alpha, color, fill, linetype, size

```
e <- ggplot(seals, aes(x = long, y = lat))
```



e + geom_segment(aes(xend = long + delta_long, yend = lat + delta_lat))

x, xend, y, yend, alpha, color, linetype, size

e + geom_rect(aes(xmin = long, ymin = lat, xmax = long + delta_long, ymax = lat + delta_lat))

xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

Twee Variabelen

X continu, Y continu

```
f <- ggplot(mpg, aes(cty, hwy))
```



f + geom_blank()

(Useful for expanding limits)



f + geom_jitter()

x, y, alpha, color, fill, shape, size



f + geom_point()

x, y, alpha, color, fill, shape, size



f + geom_quantile()

x, y, alpha, color, linetype, size, weight



f + geom_rug(sides = "bl")

alpha, color, linetype, size



f + geom_smooth(method = lm)

x, y, alpha, color, fill, linetype, size, weight



f + geom_text(aes(label = cty))

x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

Continue Bivariate Distributions

```
i <- ggplot(movies, aes(year, rating))
```



i + geom_bin2d(binwidth = c(5, 0.5))

xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size, weight



i + geom_density2d()

x, y, alpha, colour, linetype, size



i + geom_hex()

x, y, alpha, colour, fill size

Continue Functies

```
j <- ggplot(economics, aes(date, unemploy))
```



j + geom_area()

x, y, alpha, color, fill, linetype, size



j + geom_line()

x, y, alpha, color, linetype, size



j + geom_step(direction = "hv")

x, y, alpha, color, linetype, size

Fouten visualiseren

```
df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
```

```
k <- ggplot(df, aes(grp, fit, ymin = fit - se, ymax = fit + se))
```



k + geom_crossbar(fatten = 2)

x, y, ymax, ymin, alpha, color, fill, linetype, size



k + geom_errorbar()

x, ymax, ymin, alpha, color, linetype, size, width (also geom_errorbarh())



k + geom_linerange()

x, ymin, ymax, alpha, color, linetype, size



k + geom_pointrange()

x, y, ymin, ymax, alpha, color, fill, linetype, shape, size

Kaarten

```
data <- data.frame(murder = USARests$Murder, state = tolower(rownames(USARests)))
```

```
map <- map_data("state")
```

```
|<- ggplot(data, aes(fill = murder))
```



| + geom_map(aes(map_id = state), map = map) +

expand_limits(x = map\$long, y = map\$lat)

map_id, alpha, color, fill, linetype, size

Drie Variabelen

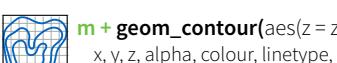
```
seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2))
```

```
m <- ggplot(seals, aes(long, lat))
```



m + geom_raster(aes(fill = z), hjust = 0.5, vjust = 0.5, interpolate = FALSE)

x, y, alpha, fill (fast)

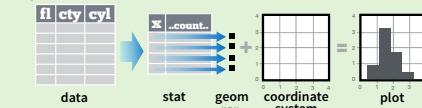


m + geom_contour(aes(z = z))

x, y, z, alpha, colour, linetype, size, weight

Stats - Een andere manier om lagen te maken

Sommige grafieken tonen een **transformatie** van de originele dataset. Gebruik een **stat** om een transformatie te tonen, bijv. `a + geom_bar(stat = "bin")`



Elke **stat** maakt extra variabelen om aan een uiterlijk te koppelen. Deze variabelen gebruiken de `..name..` syntax.

stat en **geom** functies combineren een **stat** met een **geom** om een laag te maken, `stat_bin(geom="bar")` doet hetzelfde als `geom_bar(stat="bin")`



`a + stat_bin(binwidth = 1, origin = 10)` 1D distributies
`x, y, ..count.., ..ncount.., ..density.., ..ndensity..`
`a + stat_bindot(binwidth = 1, binaxis = "x")`
`x, y, ..count.., ..ncount..`
`a + stat_density(adjust = 1, kernel = "gaussian")`
`x, y, ..count.., ..density.., ..scaled..`

`f + stat_bin2d(bins = 30, drop = TRUE)` 2D distributies
`x, y, fill | ..count.., ..density..`
`f + stat_binhex(bins = 30)`
`x, y, fill | ..count.., ..density..`
`f + stat_density2d(contour = TRUE, n = 100)`
`x, y, color, size | ..level..`

`m + stat_contour(aes(z = z))` 3 Variabelen
`x, y, z, order | ..level..`
`m + stat_spoke(aes(radius = z, angle = z))`
`angle, radius, x, xend, y, yend | ..x.., ..xend.., ..y.., ..yend..`
`m + stat_summary_hex(aes(z = z), bins = 30, fun = mean)`
`x, y, z, fill | ..value..`
`m + stat_summary2d(aes(z = z), bins = 30, fun = mean)`
`x, y, z, fill | ..value..`

`g + stat_boxplot(coef = 1.5)` Vergelijkingen
`x, y | ..lower.., ..middle.., ..upper.., ..outliers..`
`g + stat_ydensity(adjust = 1, kernel = "gaussian", scale = "area")`
`x, y | ..density.., ..scaled.., ..count.., ..n.., ..violinwidth.., ..width..`

`f + stat_ecdf(n = 40)` Functies
`x, y | ..x.., ..y..`
`f + stat_quantile(quantiles = c(0.25, 0.5, 0.75), formula = y ~ log(x), method = "rq")`
`x, y | ..quantile.., ..x.., ..y..`
`f + stat_smooth(method = "auto", formula = y ~ x, se = TRUE, n = 80, fullrange = FALSE, level = 0.95)`
`x, y | ..se.., ..x.., ..y.., ..ymin.., ..ymax..`

`ggplot() + stat_function(aes(x = -3:3), fun = dnorm, n = 101, args = list(sd=0.5))` Algemeen
`x | ..y..`
`f + stat_identity()`
`ggplot() + stat_qq(aes(sample=1:100), distribution = qt, dparams = list(df=5))`
`sample, x, y | ..x.., ..y..`
`f + stat_sum()`
`x, y, size | ..size..`
`f + stat_summary(fun.data = "mean_cl_boot")`
`f + stat_unique()`

Schalen

Scales bepalen hoe in een grafiek de data aan een uiterlijk gekoppeld wordt. Om de koppeling te veranderen gebruik je een aangepaste schaal.



Algemene Schalen

Kan bij ieder uiterlijk:
`alpha, color, fill, linetype, shape, size`

`scale_*_continuous()` - koppel continue waarden aan zichtbare waarden
`scale_*_discrete()` - koppel discrete waarden aan zichtbare waarden
`scale_*_identity()` - gebruik data waarden als zichtbare waarden
`scale_*_manual(values = c())` - koppel discrete waarden aan handmatig gekozen zichtbare waarden

X en Y locatie schalen

Gebruik met x of y uiterlijk (getoond)

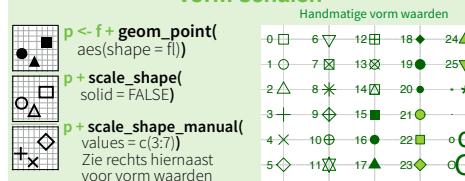
`scale_x_date(labels = date_format("%m/%d"), breaks = date_breaks("2 weeks"))` - beschouw x waarden als datum. Zie ?strptime voor label sjablonen.
`scale_x_datetime()` - beschouw x waarden als datum-tijdswaarden. Gebruik de `scale_x_date()` parameters.
`scale_x_log10()` - Plot x op een log10 schaal
`scale_x_reverse()` - Draai de x-as richting om
`scale_x_sqrt()` - Plot x op een vierkantswortel schaal

Kleur en vul schalen

Discreet Continu



Vorm schalen



Afmeting schalen



Coördinaat Systemen

`r <- b + geom_bar()`
`r + coord_cartesian(xlim = c(0, 5), ylim = c(0, 5))`

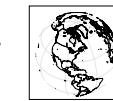
Cartesiaanse coördinaten (standaard)
`r + coord_fixed(ratio = 1/2)`

Cartesiaanse coördinaten met een vaste verhouding tussen de x en y assen
`r + coord_flip()`

Gewisselde Cartesiaanse coördinaten
`r + coord_polar(theta = "x", direction = 1)`

Polaire coördinaten
`r + coord_trans(ytrans = "sqrt")`

Getransformeerd cartesiaanse coördinaten. Geef xtrans en ytrans de naam van een functie.
`z + coord_map(projection = "ortho", orientation = c(41, -74, 0))`



projection, orientation, xlim, ylim

Kaart projecties uit het mapproj package (mercator (standaard), azequalarea, lagrange, etc.)

Positie Aanpassingen

Positie aanpassingen bepalen hoe potentieel overlappende geoms gearrangeerd worden.

`s <- ggplot(mpg, aes(fl, fill = drv))`

`s + geom_bar(position = "dodge")`

Zet elementen naast elkaar

`s + geom_bar(position = "fill")`

Stapelt elementen op elkaar en normaliseert de hoogte

`s + geom_bar(position = "stack")`

Stapelt elementen op elkaar

`f + geom_point(position = "jitter")`

Verandert de X en Y positie van ieder element een heel klein beetje om overlap te voorkomen.

Elke positie aanpassing kan hergebruikt worden als een functie met handmatige `width` en `height`.

`s + geom_bar(position = position_dodge(width = 1))`

Thema's

`r + theme_bw()` Witte achtergrond met raster
`r + theme_classic()` Witte achtergrond zonder raster

`r + theme_grey()` Grijze achtergrond (standaard thema)
`r + theme_minimal()` Minimale thema

ggthemes - Package met nog meer ggplot2 themes

Doorsneden

Doorsneden verdelen een grafiek in sub-grafieken op basis van de waarden van één of meer discrete variabelen.

`t <- ggplot(mpg, aes(cty, hwy)) + geom_point()`

`t + facet_grid(~ fl)` Doorsnede in kolommen op basis van fl
`t + facet_grid(year ~ .)` Doorsnede in rijen op basis van year
`t + facet_grid(year ~ fl)` Doorsnede in rijen en kolommen
`t + facet_wrap(~ fl)` Doorsneden in een rechthoekige layout

Gebruik `scales` om as limieten tussen de doorsneden te variëren

`t + facet_grid(~ x, scales = "free")`
 x en y as limieten worden aangepast aan de doorsneden

- `"free_x"` - alleen de x-as kan variëren
- `"free_y"` - alleen de y-as kan variëren

Gebruik `labeler` om de labels van de doorsneden aan te passen

`t + facet_grid(~ fl, labeller = label_both)`
`fl: c fl: d fl: e fl: p fl: r`

`t + facet_grid(~ fl, labeller = label_bquote(alpha ^ .(x)))`
`alpha^c alpha^d alpha^e alpha^p alpha^r`

`t + facet_grid(~ fl, labeller = label_parsed)`

`c d e p r`

Labels

`t + ggtitle("New Plot Title")`

Zet een titel boven de grafiek

`t + xlab("New X label")`

Verander het x-as label

`t + ylab("New Y label")`

Verander het y-as label

`t + labs(title = "New title", x = "New x", y = "New y")`

Gebruik schaal functies om de legenda bij te werken

Legenda

`t + theme(legend.position = "bottom")`

Zet de legenda "bottom", "top", "left", or "right"

`t + guides(color = "none")`

Zet elke legenda type voor elk uiterlijk: colorbar, legend, of none (geen legenda)

`t + scale_fill_discrete(name = "Title", labels = c("A", "B", "C"))`

Zet de legenda titel en labels met een schaalfunctie.

Vergroten

Zonder knippen (aanbevolen)

`t + coord_cartesian(xlim = c(0, 100), ylim = c(10, 20))`

Met knippen (verwijderd niet-zichtbare datapunten)

`t + xlim(0, 100) + ylim(10, 20)`
`t + scale_x_continuous(limits = c(0, 100)) + scale_y_continuous(limits = c(0, 100))`